Amendments to the Claims

This listing of claims will replace all prior versions and listings of claims in the application.

Claim 1 (original) A method of generating an intermediate representation of program code, the method comprising the computer implemented steps of:

on an initial translation of a given portion of program code, generating and storing only intermediate representation which is required to execute that portion of program code with a prevailing set of conditions; and

whenever subsequently the same portion of program code is entered, determining whether intermediate representation has previously been generated and stored for that portion of program code for the subsequent conditions, and if no such intermediate representation has previously been generated, generating additional intermediate representation required to execute said portion of program code with said subsequent conditions.

Claim 2 (original) The method according to claim 1, wherein the conditions are entry conditions, and the method comprises the computer implemented steps of:

generating an Intermediate Representation Block (IR Block) of intermediate representation for each Basic Block of program code as it is required by the program, each IR Block representing a respective Basic Block of program code for a particular entry condition;

storing target code corresponding to each IR Block; and

when the program requires execution of a Basic Block for a given entry condition, either:

(a) if there is stored target code representing that Basic Block for that given entry condition, using said stored target code; or

(b) if there is no stored target code representing that Basic Block for that given entry condition, generating a further IR Block representative of that Basic Block for that given entry condition.

Claim 3 (original) The method of claim 1, wherein the intermediate representation of the program code is generated dynamically as the program code is running, the method comprising the computer implemented steps of:

at a first iteration of a particular subject code instruction having a plurality of possible effects or functions, generating and storing special-case intermediate representation representing only the specific functionality required at that iteration; and

at each subsequent iteration of the same subject code instruction, determining whether special-case intermediate representation has been generated for the required functionality required at said subsequent iteration and generating additional special-case intermediate representation specific to that functionality if no such special-case intermediate representation has previously been generated.

Claim 4 (previously amended) The method according to claim 3, wherein the said special-case intermediate representation is generated and stored and an associated test procedure is generated and stored to determine on subsequent iterations of the respective subject code instruction whether the required functionality is the same as that represented by the associated stored special-case intermediate representation, and, where additional special-case intermediate representation is required, an additional test procedure associated with that special-case intermediate representation is generated and stored with that additional special-case intermediate representation.

Claim 5 (previously amended) The method according to claim 4, wherein the additional special case intermediate representation for a particular subject code instruction and the additional associated test procedure is stored at least initially in subordinate relation to any existing special-case intermediate representation and associated test procedures stored to represent the same subject instruction, such that upon the second and subsequent iteration of a subject code instruction, a determination of whether or not required special-case intermediate representation has previously been generated is made by performing said test procedures in the order in which they were generated and stored until either it is determined that special-case intermediate representation of the required functionality exists, or it is determined that no such

required special-case intermediate representation exists in which case more additional intermediate representation and another associated test procedure is generated.

Claim 6 (previously amended) The method according to claim 5, wherein the intermediate representation is optimised by adjusting the ordering of the test procedures such that a test procedure associated with a more frequently used special-case intermediate representation is run before a test procedure associated with a less frequently used special-case intermediate representation rather than ordering the test procedures in the order in which they are generated.

Claim 7 (original) The method of claim 1, comprising translating the program code written for execution by a processor of a first type so that the program code may be executed by a processor of a second type, using the generated intermediate representation.

Claim 8 (original) The method according to claim 7, wherein said translation is dynamic and performed as the program code is run.

Claim 9 (original) The method of claim 1, comprising optimising the program code by optimising said intermediate representation.

Claim 10 (original) The method according to claim 9, wherein the method is used to optimise the program code written for execution by a processor of a first type so that the program code may be executed more efficiently by the processor.

Claim 11 (original) A method for generating an intermediate representation of program code written for running on a programmable machine, said method comprising:

(i) generating a plurality of register objects for holding variable values to be generated by the program code; and

(ii) generating a plurality of expression objects representing fixed values and/or relationships between said fixed values and said variable values according to said program code;

said intermediate representation being generated and stored for a block of program code and subsequently re-used if the same block of program code is later re-entered, and wherein at least one block of program code can have alternative un-used entry conditions or effects or

functions and said intermediate representation is only initially generated and stored as required to execute that block of program code with a then prevailing set of conditions.

Claim 12 (original) A method according to claim 11, wherein for a given block of program code, it is determined whether a previously stored intermediate representation therefor was for the same now currently prevailing set of conditions and, if not, then generating and storing additional intermediate representation as required to execute the block of program code for the new now currently prevailing set of conditions.

Claim 13 (previously amended) A method of generating a target code representation of program code, the method comprising the computer implemented steps of:

on an initial translation of a given portion of the program code, generating and storing only target code which is required to execute that portion of program code with a prevailing set of conditions; and

whenever subsequently the same portion of program code is entered, determining whether target code has previously been generated and stored for that portion of program code for the subsequent conditions, and if no such target code has previously been generated, generating additional target code required to execute said portion of program code with said subsequent conditions.

Claim 14 (previously amended) A method of dynamically translating first computer program code written for a first programmable machine into second computer program code for running on a different second programmable machine, said method comprising:

(a) generating an intermediate representation of a block of said first computer program code;

(b) generating a block of said second computer program code from said intermediate representation;

(c) running said block of second computer program code on said second programmable machine; and

(d) repeating steps a-c in real time for at least the blocks of first computer program code needed for a current emulated execution of the first computer program code on said second programmable machine.

Claim 15 (original) A system for generating an intermediate representation of program code comprising:

means for generating and storing, on an initial translation of a given portion of program code, only intermediate representation which is required to execute that portion of program code with a prevailing set of conditions; and

means for determining, whenever subsequently the same portion of program code is entered, whether intermediate representation has previously been generated and stored for that portion of program code for the subsequent conditions, and if no such intermediate representation has previously been generated, generating additional intermediate representation required to execute said portion of program code with said subsequent conditions.

Claim 16 (original) A system for generating an intermediate representation or program code written for running on a programmable machine, the system comprising:

means for generating a plurality of register objects for holding variable values to be generated by the program code; and

means for generating a plurality of expression objects representing fixed values and/or relationships between said fixed values and said variable values according to said program code; and

means for generating and storing intermediate representation, said intermediate representation being generated and stored for a block of program code and subsequently re-used if the same block of program code is later re-entered, and wherein at least one block of program code can have alternative un-used entry conditions or effects or functions and said intermediate representation is only initially generated and stored as required to execute that block of program code with a then prevailing set of conditions.

Claim 17 (previously added) The method of claim 1 wherein said intermediate representation of program code is generated at run time.

Claim 18 (previously added) The method of claim 11 wherein said intermediate representation of program code is generated at run time.

Claim 19 (previously added) The method of claim 13 wherein said target code representation is generated at run time.

Claim 20 (previously added) The system of claim 15 wherein said intermediate representation of program code is generated at run time.

Claim 21 (previously added) The system of claim 16 wherein said intermediate representation of program code is generated at run time.

Claim 22 (previously added) The method of claim 1 wherein said steps are performed at run time.

Claim 23 (previously added) The method of claim 11 wherein said plurality of register objects and plurality of expression objects are generated at run time.

Claim 24 (previously added) The method of claim 13 wherein said steps are performed at run time.

Claim 25 (previously added) The system of claim 15 wherein the function of generating and storing on an initial translation, the function of determining, and the function of generating additional intermediate representation are each performed at run time.

Claim 26 (previously added) The system of claim 16 wherein the functions of generating a plurality of register objects, generating a plurality of expression objects, and generating and storing intermediate representation are each performed at run time.

Claim 27 (new) The method of claim 1, wherein whenever subsequently the same portion of program code is entered, the method comprises the computer implemented steps of:

determining whether said subsequent conditions are different from said prevailing set of conditions, and

if different, generating and storing a different intermediate representation for said same portion of program code required to execute said portion of program code with said subsequent conditions.

Claim 28 (new) The method of claim 13, wherein whenever subsequently the same portion of program code is entered, the method comprises the computer implemented steps of:

determining whether said subsequent conditions are different from said prevailing set of conditions, and

if different, generating and storing different target code for said same portion of program code required to execute said portion of program code with said subsequent conditions.

Claim 29 (new) The method of claim 11, wherein said at least one block of program code has a plurality of possible alternative entry conditions or effects or functions,

further wherein said intermediate representation is only initially generated and stored to represent only a portion of said plurality of possible alternative entry conditions or effects or functions as required to execute that block of program code with said prevailing set of conditions.

Claim 30 (new) The method of claim 29, wherein said intermediate representation does not represent any un-used entry conditions or effects or functions from said plurality of possible alternative entry conditions or effects or functions.